福井大学
工学部　研究報告
第46巻　第2号
1998年9月

UNIXによるMIDI Daemonの開発

青山 大介*　浅井 祥隆*　山東 俊喜*
長谷川 武光**　佐藤 義雄**

Making a MIDI Daemon on UNIX Operating System

Daisuke AOYAMA, Yoshitaka ASAI, Toshiki SANDO,
Takemitsu HASEGAWA and Yoshio SATO

(Received Aug. 28, 1998)

A MIDI controlable system is constructed to provide a music facilities on UNIX-Workstation (WS). The system constructed is a MIDI Daemon for a variety of interfaces to music application programs. Users can usethe present system to enjoy music on UNIX-WS in the same was as on PC.

Key Words : MIDI, Daemon, UNIX-WS, Network

1　はじめに

近年コンピュータの使用される分野は数多く、とりわけ音楽の分野における楽曲の演奏はその顕著な一例であろう。楽器を使っての演奏ではなく、曲のデータをコンピュータへ入力し、電子音源を使い演奏する音楽は様々な所で耳にすることが出来る。これらはDTM(Desk Top Music)と呼ばれる。その名の通り机上にて作り出される音楽であり、最近では、演奏に限らず作曲までもがコンピュータ上で行なわれている。そしてコンピュータにおける音楽の演奏・作曲環境は日々向上し、更なる進歩が期待されている。

DTMの多くはパーソナルコンピュータ上でのみ提供され、ワークステーション上ではその応用例に乏しいのが現状である。本講座で構築・改良されてきたUNIX上のMIDI制御システムは、発足当時の

*工学研究科情報工学専攻 **工学部情報工学科

単一モジュール・シングルプロセスから、クライアント・サーバ形式を取り込んだマルチプロセス化と優良なインタフェースを提供するまでに至った。

UNIX の提供する重要な特徴であるネットワーク環境は、MIDI で扱われるデータの送受信にも多面で有用性があり、UNIX での実現意義の確立において非常に有益であるが、本システムでは有効に活用されていなかった。また、クライアントへ提供するライブラリ (ツール) が十分に用意されておらず、サーバ側の機能強化についても急務の問題となっている。

本研究では、UNIX での通信機能を付加した、実用的な MIDI 制御システムの開発を行う。また、最終的にサーバを OS の起動時に実行させ、MIDI daemon として機能させることを目的とする。

2 MIDI 規格について

MIDI(Musical Instrument Digital Interface の略) は、国内外のメーカーでの国際的な規格である MIDI 規格を根底とする、楽器の演奏情報をデジタルで通信するためのインターフェイスである。MIDI 規格は音楽のデジタル通信法の一つであるが、通信線に楽器音等のサンプリング情報を流し、そのまま音に復元する方式ではない。演奏の情報 (音を発するタイミング・音にかける効果・音自体の音量や長さ等の音楽的譜面情報) を転送するものであり、発する音素は受信側へ依存する事になる。

MIDI 規格によって定義されたデータ通信の方法に従い、演奏情報の入出力を行なっている電子楽器であれば、国内外の機種を問わず自由に接続し通信する事が出来る。通常の形態としては MIDI-IN と MIDI-OUT の 2 本のケーブルによって情報のやりとりを行うが、情報のみを一方的に送信して自動演奏を行なう事も DTM 等でしばしば見られる。

コンピュータが MIDI においてどんな働きを期待されるかを示すと、演奏データの快適な入力・加工・編集や、各種演奏データの表示による音楽の視覚化、曲データファイルの編集による整理、各種 MIDI 機器とのやりとり・データ送信方向の切替え、付加効果による高度な演奏等、多岐に渡る。

3 昨年度までの研究の概要と問題点

3.1 RS-MIDI インターフェイス

研究室内のワークステーション SONY NEWS3860 からの出力をそのまま MIDI 機器に転送することは出来ない。これは、NEWS3860 がハードウェアでサポートしている転送レートでは、MIDI 機器がデータを受信できないことに起因する。これを解決するには、外部からクロック信号を与え、MIDI 機器との同期を取る事の出来るインターフェイスを別途提供する必要がある。平成 6 年度の研究では RS-MIDI インターフェイスを作成し、ワークステーション上で MIDI 制御システムが実現できる体制となった。

但し、インターフェイスとして MIDI 機器と接続するのはホストコンピュータであり、端末に直接接続した MIDI 機器は想定していない。この問題については、端末での MIDI 機器の接続、設定に関する資料に乏しい為、本研究では扱っていない。

3.2 MIDI 演奏ドライバ

平成 7 度の研究では、平成 6 年度のドライバを改善し、グラフィカルユーザインタフェースの搭載、MIDI ファイルの管理機能を追加する改良が行なわれた。[3] しかしグラフィカルユーザインターフェイスの搭載と演奏情報の視覚的表示は、そのグラフィック処理による負担が演奏処理に影響を及ぼすという欠点があった。昨年度の研究では MIDI データ処理をグラフィック描画処理から切り離し、クライアント・サーバ形式による MIDI ドライバを開発し、MIDI 処理に対する問題を解消した [4]。

これらの研究成果によって、ワークステーション上での MIDI の処理環境が構築された訳だが、その OS である UNIX 上で実現するメリットの薄いことを指摘されており、また、MIDI ツールの開発環境

としての実用的なクライアント・サーバ形式の整備・強化も新たに問題として浮上することとなった。これらの問題を解消することが本研究の当初の課題であった。更に、昨年度の研究における課題となっていたライブラリの開発についても考慮し、クライアントへの機能提供案として、システムの改良のための再設計を行なっている。

4 MIDI管理サーバ

UNIXの特長であるネットワーク機能を取り入れた、MIDIアプリケーション開発を支援するMIDI制御環境ソフトウェアシステムを構築する。スタンダードMIDIファイルに含まれるMIDIメッセージの他、直接MIDIに関係しないデータ等様々なメッセージの送受信や処理を行うのがMIDI管理サーバ(以下、本システム)である。本章では、本システム開発における過程について述べる。

4.1 本システムの参照モデルへの位置付け

本システムはUNIXで提供される種々のシステムコールを利用する。特にネットワーク機能で利用するシステムコールは、プロセス間のトランスペアレントなデータ転送までであり、本システムでは、データ送受信制御・同期制御、構造体データの入力・送受信・表示・制御機能を実現する。これをOSI参照モデルに対応させると第4層(トランスポート層)までがシステムコールの機能であり、本研究では第5層(セッション層)、第6層(プレゼンテーション層)の機能を実現することになる[1]。

4.2 昨年度MIDI演奏ドライバの課題

MIDI制御システムのクライアント・サーバ化は昨年の研究により実現された。しかし、クライアントへ提供するライブラリ(ツール)が十分に用意されておらず、サーバの機能が明瞭でなかった。また、メッセージを送信するクライアントがサーバとソースコード上で結合しており、サーバ自体が実験を兼ねたものであった。以上より、実用的な開発環境として更なる設計・開発の余地が存在した。

昨年度のドライバで挙げられた展望では、ネットワークを活用した機能の追加がある。このネットワーク活用案として2つの方策が与えられた。その1つに、異なるホストでのMIDIドライバ間によるデータ通信機能があった。これは、MIDIドライバ上で直接MIDIファイル、もしくはMIDIメッセージをネットワークを利用して送受信する。ワークステーションを利用するに当たり、ネットワークの利用は避けられないものであり、データの共有、複製等は無意識的に行なっている。MIDI制御システムへのネットワーク機能の付加はその用途を広める事が期待出来る。MIDIファイルの転送・複写等は、本システムへの導入を行なわずともターミナル等でファイル操作に関するコマンドを実行することで可能である。しかし、MIDIメッセージのホスト間送受信、他ホストのMIDIファイル・メッセージの管理は直接には不可能である。ネットワーク機能を応用する事でこれらを実現することが出来、MIDI制御システムを個人利用の域から脱する事が可能となる。また、MIDI制御に特化したネットワーク機能を提供する事により、繁雑な手続きを不要にし、ネットワークを利用したMIDIを扱うアプリケーションが容易に開発出来る。

以上の事から、ドライバの改良とクライアントへ簡便に利用可能なライブラリの提供・拡充が必要であり、MIDI制御に特化したネットワーク機能の追加が要求される。

4.3 サーバ構成の改善

クライアント・サーバ方式を取り入れることによって、ツールの要求を受け入れるドライバは、そのメッセージ処理に対して的確な動作が要求されることになる。例えば、ツールのいい加減なメッセージ送信時におけるサーバの振舞規定、サーバへのメッセージ送信が複数同時に起こった場合の対処、許容範囲数以上のメッセージ送信時の制御等、重要かつ必要な機能が挙げられる。

このような事象に対してサーバとして機能しなくなる状況を排除し、かつ、新たにネットワーク機能を取り入れる為、本システムではサーバ側の大幅な構成変更を行なった。

4.3.1 昨年度システムの設計案の採用

次に、昨年度システムの構成図(図1)と本システムへ継続して採用する設計案の一部を示す[4]。

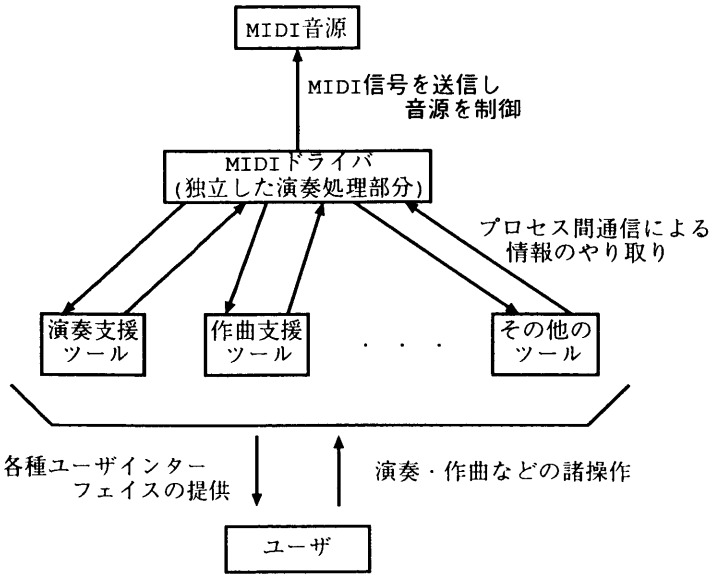


図 1: 昨年度 MIDI 制御ドライバ構成図

MIDI 音源を MIDI ドライバ(前述の独立した演奏部分)が制御し、MIDI ドライバへ各ツールが要求を発する。演奏処理を独立させたので、MIDI 音源を直接制御しているプロセスは MIDI ドライバのみになっている。

各ツールは、ドライバとユーザを仲介するインターフェイスとしての役割を果たす。具体的には、平成7年度研究のグラフィカル MIDI ドライバの様な、演奏情報の表示機能・ファイラ機能等を提供するものである。演奏支援だけでなく、例えば、譜面入力エディタの様な作曲支援の環境なども含むことが出来る。各ツールはユーザの操作等に応じてドライバへ要求を発する。ドライバは各ツールが必要としている情報を提供する。これらの間で双方向のプロセス間通信を行ない、情報をやり取りする。

プロセス間通信方式には様々なものが存在する。「MIDI ファイル全体をバッファに保存しておく必要があるプロセスは MIDI ドライバのみである」、「MIDI ファイルは符号無文字型のバイナリファイルとみなすことが出来る」等の理由から、メッセージキューシステムコールが用いられている。

送られるメッセージは、各ツールからドライバに送られる要求と、ドライバから各ツールに送られる情報の2つに大別される。前者は、ツール毎に区別する必要は無いが、後者は、各ツールごとに独立して送らなければならない。このため単一のメッセージキューによるメッセージの多重化が用いられる。

メッセージの多重化は、メッセージタイプと呼ばれる long 型のパラメータを用いる事によって行なう。これにより、単一のメッセージキューから特定のメッセージを選んで受け取ることが可能になる。概要を図2に示す。

ドライバから各ツールに対してのメッセージは、それぞれのツール毎に区別するユニークなメッセージタイプを持たせる必要がある。これについては各ツールのプロセス ID を用いることで解決している。

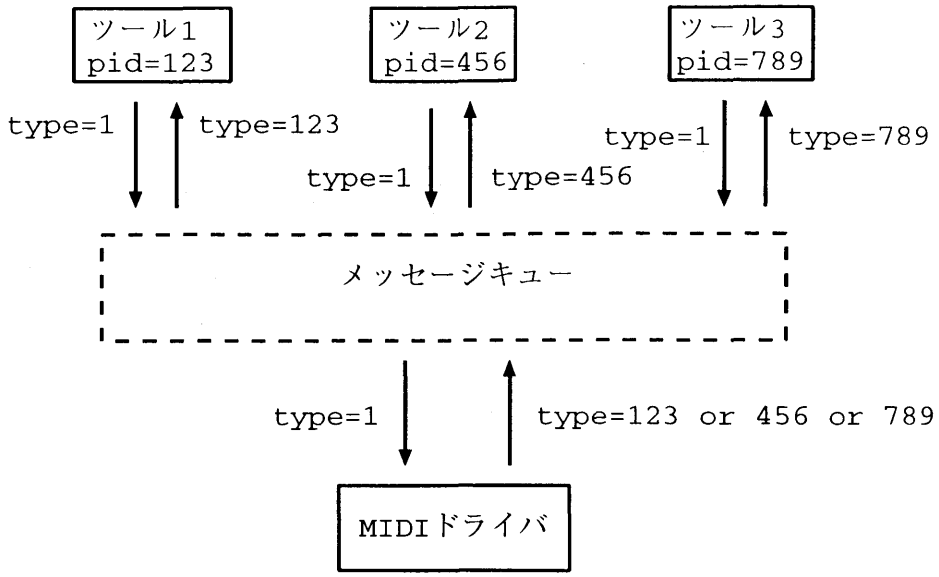


図 2: 単一のメッセージキューによるメッセージの多重化

4.3.2 本システムの基本構成

本システムは、主に 4 種類のプロセス (PMS、DAS、DPS、クライアント) から成り立っている。その内、3 つのプロセス (PMS、DAS、DPS) が本システムのサーバに相当する。これらのプロセスが相互にメッセージ、MIDI データのやり取りを行う。メッセージについての詳細は「メッセージ処理」の節で説明する。図 3 に、本システムの実行状態における基本構成図を示す。

以下に各プロセスの機能を述べる (各サーバのネットワーク機能等詳細については次節で述べる)。

- PMS (Port Mapping Server)

  ホスト間のメッセージ通信を行う際に必要な情報を管理するサーバであり、サーバグループ内に唯 1 つ存在する。DAS が他ホストへ送受信するメッセージ・データのルータとして機能する。詳細については「ネットワーク機能」の節で述べる。

- DAS (Data Administration Server)

  各ホスト上毎に唯 1 つ起動されるサーバであり、ホスト内に存在するクライアント・DPS へのメッセージ・データの送受信、MIDI データバルクの管理、メタイベントメッセージの処理、他ホスト内 DAS へのメッセージ・データの送受信等を行う。起動時に PMS へ自身の起動情報 (ホスト名等) を登録する。また、PMS を経由して他ホストへメッセージ送信する。

- DPS (Data Playing Server)

  各ホスト上毎に唯 1 つ起動されるサーバであり、主として DAS より渡される MIDI データの MIDI 機器へ送信、MIDI 機器送信中の MIDI メッセージ内メタイベント等の DAS への処理依頼、クライアントからの単純 MIDI メッセージ受信を行う。単純 MIDI メッセージとは、MIDI ファイルとして DAS に格納されず、クライアントと直接通信可能な、数バイト程度の MIDI メッセージである。MIDI 機器の初期化メッセージ送信、モニタリング、クライアン

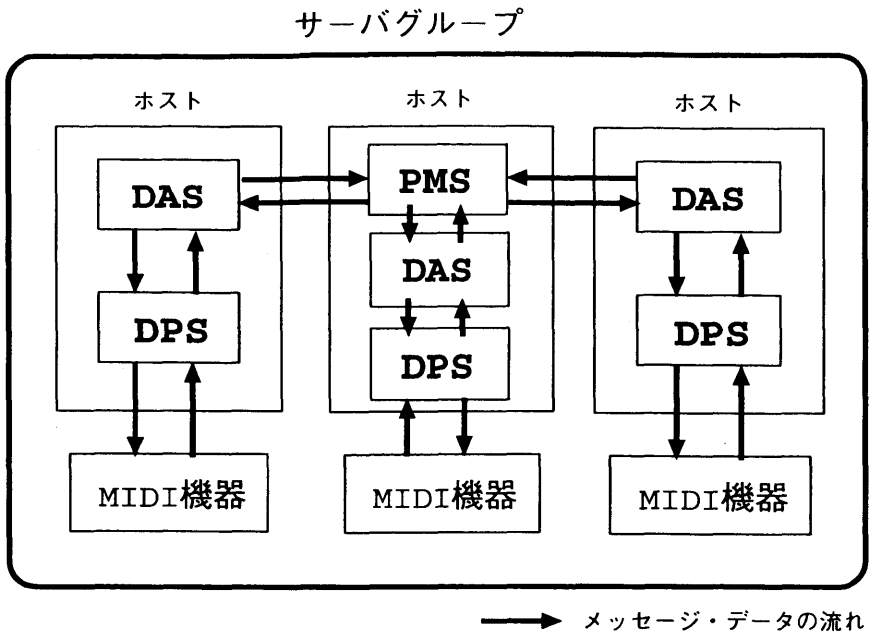


図 3: 実行状態における基本構成図

トからのリアルタイム MIDI メッセージ送信等に利用する。なお、DPS は昨年度に開発された MIDI ドライバを本システム上で扱う機能、メッセージに対応させたものである。

- クライアント

  DAS へ要求メッセージを発するプロセスであり、一般にサーバへ接続するユーザプロセスがこれにあたる。クライアントは所定のライブラリをインクルードした後、サーバへ接続する必要がある。昨年度システムのツールはクライアントを包含したもので、ユーザ、もしくはアプリケーションへの直接のインターフェイスとなる。

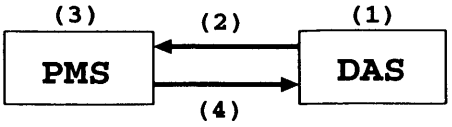
4.3.3 ネットワーク機能

本システムで扱われる様々な情報は、メッセージキューを用いて各サーバやクライアントへ受渡しを行なう。これは同一ホスト間においてのみ有効であり、別ホストで動作するサーバへの送受信手段を与えるものではない。従って、メッセージキュー単体ではホスト間を繋ぐネットワークを利用する事は出来ない。本システムでは新たにソケットと呼ばれるデータ送受信法を組み込んだ。ソケットは UNIX において一般化されたデータ送受信法の 1 つであり、メッセージキューと類似したプロセス間通信以外に、広範なネットワーク上のデータ通信法を提供するシステムコール群である [2]。ソケットは大別して 2 種類に分類され、プロトコルと通信範囲によって UNIX ドメインと INET(InterNET) ドメインとに呼び分けられる。UNIX ドメインは単一の UNIX システム内を領域とする通信法であり、INET ドメインは IP アドレス等を用いた、ネットワーク規模で利用可能な通信法である。本システムではネットワークによるホスト間通信を実現する為、後者を利用した。

INET ドメインは、internet 上で必要なパラメータを用いて接続を行なう。しかし、相手を指定する際に数種類の情報が必要になり、また、それらの情報を直接知る手段が与えられない。従って、各サー

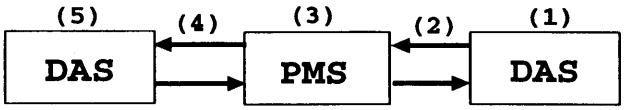
バが相互の情報を常に全て把握することは困難である。この問題を解決する為、全てのサーバ情報を保持・管理するサーバを新たに定義し、これをPMSと命名した。DASは各ホストにて起動する際、PMSへDAS自身のソケット通信で利用する為のパラメータを与える。PMSはこれを内部に保持し、DASによるホスト間通信の際に利用する。ホスト間通信の際、DASはPMSへ他ホストDASへ送信するメッセージを渡す。PMSは送信先のDASのホスト情報を検索し、存在する場合、保持する送信先ホストのパラメータを用いて、メッセージを相手DASへ送信する。

DASのPMSとの接続手順図を図4、図5に示す。



(1) DAS起動、ソケット接続準備
(2) PMSへ接続、DASホスト情報をPMSへ送信
(3) 受け取ったホスト情報を内部に格納
(4) ホスト情報格納通知を送信

図4: PMSへDASのホスト情報を与える場合

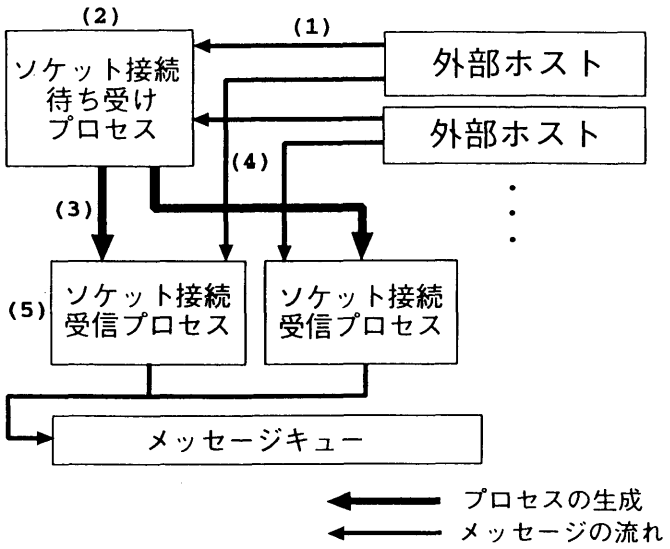


(1) ソケット接続準備
(2) PMSへ接続、DASメッセージ送信
(3) 送信先ホスト情報検索、接続情報付加
(4) 相手先DAS接続、メッセージ送信
(5) メッセージ受信

図5: ホスト間DASのメッセージ通信の場合

以上の手順はPMSをホスト間DASメッセージ通信のルータとして利用することを示している。ホスト間通信メッセージを全てPMSを経由させる事で通信ホスト間の経路自体は冗長になるが、DASへPMS内の情報を与える必要が無くなり、サーバ単体毎のソケット通信処理によるオーバヘッドを抑えることが可能である。

ソケット通信ではメッセージキューの様な多重化の技法は利用出来ない。よって、複数同時の受信接続を行なう為にプロセス多重化を応用した(図6)。これは、ソケット通信接続要求を受けるプロセスをDAS内に用意し、接続要求が来た場合に接続受信用のプロセスを生成し、受信を行なうものである。受信メッセージはメッセージキューへ移され、目的先へ送信する。送信元との接続が終了した場合は、生成された接続受信プロセスは終了される。



(1) 外部ホストからのソケット接続要求
(2) 接続要求を受信
(3) ソケット受信用プロセス生成
(4) ソケット接続受信プロセスへメッセージ送信
(5) メッセージをメッセージキューへ格納

図 6: 同時受信時のプロセス生成

ネットワーク機能を中心に表したPMS、DASの構成図をそれぞれ図7、図8に示す。

ソケット送信でも受信時と同様に送信用プロセスが存在するが、DASからのソケット送信時は受信時と異なり、同時に1つの送信のみを行なう。これは、DAS内でのメッセージ処理は同時に1つであり、複数の同時送信はありえないためである。

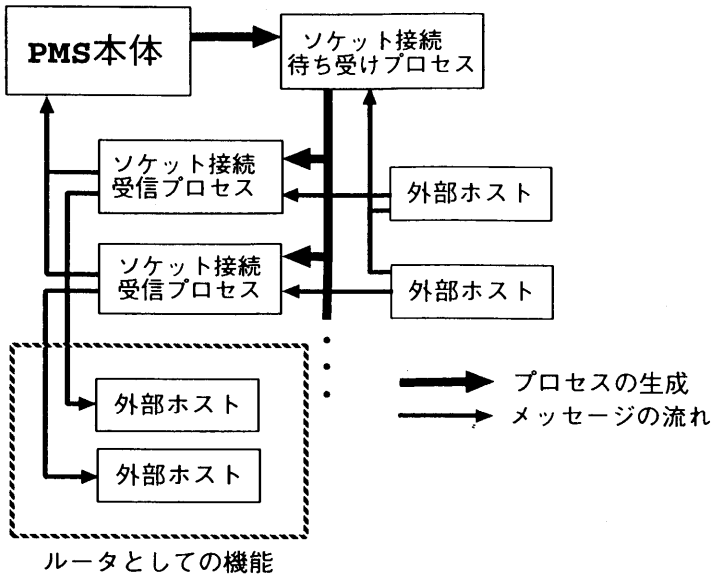


図 7: PMS の構成

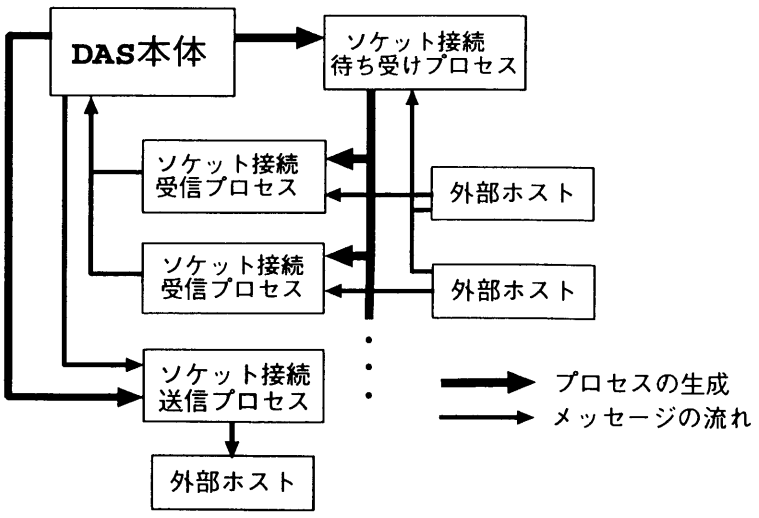


図 8: DAS の構成

4.3.4 メッセージ処理

メッセージの具体的構成を説明する(メッセージ構造体の詳細は付録で述べる)。本システムで用いるメッセージは、送受信用情報、要求内容、パラメータ等を要素とした、型定義によって示される構造体である。これをメッセージキューとネットワーク機能で共用する。

前節にて述べたメッセージキュー利用法は、long 型を送信先のユニーク値、即ちプロセス ID を与えることで多重化を実現している。しかし、ネットワークによるホスト間メッセージでプロセス ID はユニーク値とは成りえない為、事前にホスト名の参照によってメッセージのホスト内外の区別認識を行なう。

サーバのユニーク値は定数を用いる。クライアント同様にプロセス ID の利用ではサーバのプロセス ID を何らかの手段で公開する必要がある。予めユニークな定数値を用いる事で、クライアント内のソースコードへ直接与える事が可能である。

クライアントはメッセージをメッセージキューに格納する事でサーバへ要求を伝える。サーバからクライアントへの場合、実用上メッセージキューを用いる事は出来ない。理由として 2 つの問題が挙げられる。

1 つはクライアントの振舞の問題である。基本的に、クライアントが異常な動作を行なう可能性がある。異常がサーバへ影響を及ぼす事は避けねばならない。サーバからクライアントへのメッセージをメッセージキューに格納する場合は、クライアントがメッセージを取り出さなくなる事によりメッセージが蓄積し、キューの許容量を圧迫する。2 つ目はメッセージキューの仕様の問題である。キューに格納出来るメッセージのサイズ・数制限は比較的小さく、多数のクライアントが動作する状況で送受信両方を頼らせることが困難である。

従って以上 2 点の問題に対処する為、クライアントのメッセージ受信をプロセス間通信の 1 つである共有メモリ上で行なう事にした。共有メモリであればメッセージキューと独立して利用出来、メモリのサイズを十分に取る事が出来る。図 9 に、本システムのホスト単位におけるメッセージの流れを示す。

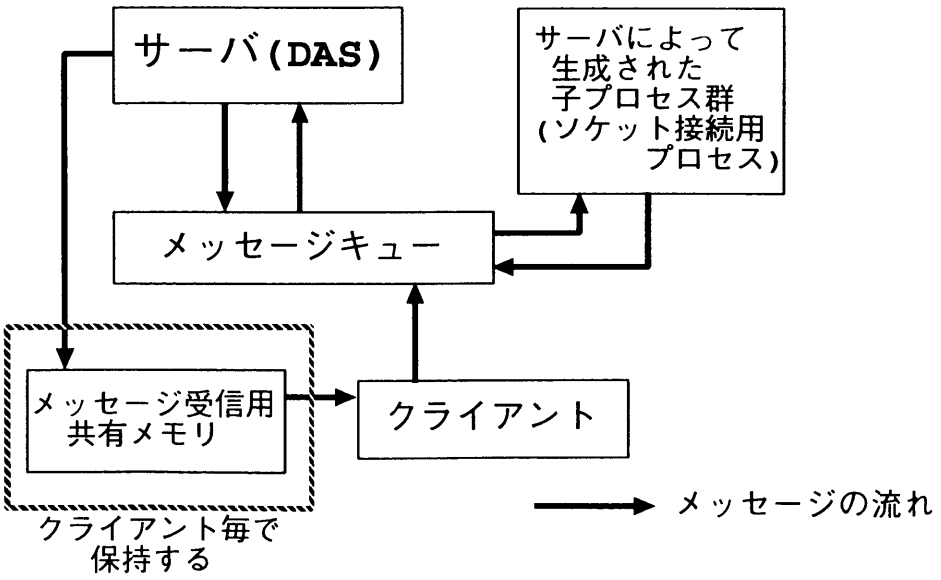


図 9: メッセージの流れ

クライアントのメッセージ受信法と送信に関する詳細は「クライアント」の章で述べる。

4.3.5 サーバの安定化

クライアントの様々な要求への随時応答や実用性を高めるため、サーバの安定した実行が必要である。本システムでは安定化を目的として以下の設計案を実装した。

- エラーメッセージ送受信機能
- 同一領域への複数プロセスのアクセスに対する各種排他処理
- 子プロセス再生成機能・サーバ再起動機能
- サーバ実行状況モニタ用クライアント

5 クライアント

サーバはクライアントから要求を受ける事が出来るが、サーバのみの開発ではクライアントはサーバへメッセージを送る手段を持たない。サーバを利用する為の何らかのインタフェースをクライアントの開発側へ与える必要がある。本節では、クライアントから見た各サーバの利用法について説明する。

5.1 メッセージ受信

クライアントとサーバのメッセージ授受の流れを図 10 に示す。クライアントのメッセージ受信は、前節で述べた理由からメッセージキューは用いない。
代理のプロセス間通信法として共有メモリを利用している。共有メモリはメッセージキューの様な FIFO バッファやメッセージタイプによるメッセージ識別機能を備えておらず、また排他処理等のアクセス制御も行なわない。代わりに比較的大きいメモリ空間を高速にアクセスする事が可能である。この特徴に対して、共有メモリへサーバから受信するメッセージを順番に格納する FIFO バッファと、サーバとクライアントが同時にアクセスする場合の排他処理機能を付加する事で、クライアントが受信専用のメッセージキューとして利用する事を可能にした。メッセージ受信用共有メモリはクライアント各々が自身のプロセス ID をキーとして割当てを行ない、メッセージ内のクライアント PID を利用してアクセスする。つまり、メッセージタイプを共有メモリアクセスのキーに利用する事でメッセージキューと共有メモリの併用を可能にしている。

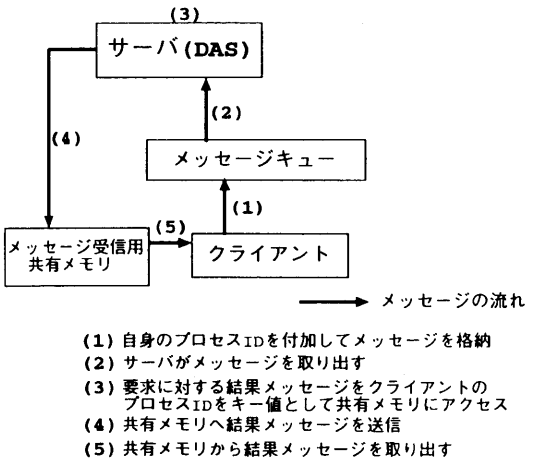


図 10: クライアント - サーバ間のメッセージの流れ

5.2 単純 MIDI メッセージ

単純 MIDI メッセージとは基本構成の節で述べた動作、即ちクライアントが DPS へ要求を発する為に与えられる、主として数バイト程度の MIDI メッセージである。

単純 MIDI メッセージを格納するバッファは、サーバとの接続手続を行なった際に共有メモリのアドレスとして渡される。このアドレスへ MIDI メッセージと書き込みバイト数を代入することでサーバにメッセージを送信する事が出来る。

5.3 ライブラリの提供

クライアントがサーバへメッセージを送信するという動作だけでも、複数のメッセージ処理手続や変数の利用を要求する。また、メッセージの要求内容によって複数の動作が一続きとなる場合も存在する。このようなクライアントが受け持つ繁雑な処理の記述を減らし、かつ不正なメッセージ送受信を防止する為に、サーバ接続専用のライブラリヘッダを作成し、サーバとのインタフェースとして提供した。ライブラリヘッダには、クライアントがサーバとの接続に必要な各種変数、型宣言、定数宣言、サーバと兼用するヘッダのインクルード、前処理、関数等が記述されている。クライアントは自身のコード上にライブラリヘッダをインクルードする事で、サーバとのメッセージ通信機能を利用出来る。本研究では、このヘッダを利用してクライアントのサンプルを作成している。

クライアントのサンプルについては、次節にて述べる。

6 実行方法と実行結果

本章では実際に MIDI 管理サーバを使用する際の、各サーバの起動と MIDI データの送信・再生、DAS 内 MIDI データのコピー、終了法についての実行例を示す。また、本システムは研究室のハードウェア構成上、sgauss(NEWS3860) でのみ MIDI 出力が可能な為、ネットワーク機能を利用する際は sgauss 以外のホストから操作している。

6.1 初期設定、PMS 起動

まず、それぞれのホストマシンの背後にある RS-232C 端子と RS-232C インターフェイスの RS-232C 端子、MIDI 機器の背後にある DIN5 ピンとインターフェイスの DIN5 ピンを各々繋ぐ。MIDI 機器側の MIDI メッセージ受信準備も行なっておく。

PMS を常駐させるホストを選択し、PMS を起動する。すると、起動メッセージを表示し、PMS 実行ファイルの存在するディレクトリに起動状況ファイルを出力し、ホストに常駐する (図 11)。

```
sgauss%pms &
[1] 1160

<PMS>ver 0.9 Port Map Server for MIDI daemon
sgauss%programmed by blue-mo@fukui-u, 1997-1998


sgauss%ps
  PID TT STAT TIME COMMAND
 1132 e  S    0:00 -csh (tcsh607)
 1160 e  S    0:00 pms
 1161 e  S    0:00 pms_skwt 1160 3000
 1162 e  R    0:00 ps
sgauss%
```

図 11: PMS の常駐

6.2 DAS 起動

DAS を起動するホストを選択し、DAS を起動する。図 12 に示す様に PMS と同様に起動メッセージを表示し、DAS 実行ファイルの存在するディレクトリに起動状況ファイルを出力し、ホストに常駐する。この時、PMS 起動状況ファイルを読み込む。また、DAS 自身が起動したホスト情報を PMS へ登録する為、PMS へ接続する際の送受信プロセスがホスト接続メッセージを表示する。PMS が常駐するホストでも DAS を起動させる事が出来るが、他ホストで PMS が起動している場合は起動状況ファイルを参照する際、そのホストが NFS 等によってファイルアクセス可能でなければならない。

```
agauss%das &
[1] 26509
<DAS>ver 0.6 Data Administration Server for MIDI daemon
programmed by blue-mo@fukui-u, 1997-1998

DAS 起動状況出力中...
agauss%
PMS 起動状況ファイル読みだし中...

DAS : connect from sgauss.

agauss%ps
  PID TT STAT TIME COMMAND
29198 2  S    0:00 -tcsh(blue-mo from agauss) (tcsh607)
26226 2  S    0:00 das
26227 2  S    0:00 das_sksnd 29226
26228 2  S    0:00 das_skwt 29226 3000
26229 2  S    0:00 das_skrcv 29226 4 sgauss
26230 2  R    0:00 ps
agauss%
```

図 12: DAS の常駐

6.3 DPS 起動

図 13 の様に DAS が存在するホストそれぞれに DPS を起動し、常駐させる。起動メッセージが現れ、DPS 内部では各種初期設定、他サーバへの接続、RS-232C デバイスの設定・MIDI 機器の初期化を行なう。

```
agauss%./xm &
[2] 29377

<DPS(xm)> for MIDI control with RS-232C device
programmed by blue-mo,pumpkin@fukui-u, 1996-1998
agauss%


agauss%ps
  PID TT STAT TIME COMMAND
29198 1  S    0:00 -tcsh(blue-mo from sgauss) (tcsh607)
29226 1  S    0:00 das
29227 1  S    0:00 das_sksnd 29226
29228 1  I    0:00 das_skwt 29226 3000
29229 1  I    0:00 das_skrcv 29226 4 sgauss
29377 1  R    0:04 ./xm
29381 1  R    0:00 ps
agauss%
```

図 13: DPS の常駐

6.4 サンプルクライアントの実行

本システムの機能を利用する為のサンプルクライアントを実行する。するとメニューが表示され、入力を待つ状態になる(図 14)。メニューの先頭に示されている番号を入力する事で各動作を実行する。

まず、MIDI ファイルの読み出しを行なわせる。図 15 に示す様に番号 150 を入力し、送信するホスト名を入力する。次に、MIDI ファイル名を入力する事で、サーバへ MIDI ファイルを送信する。ここで、該当するホストに DAS が起動されていない場合は PMS からエラーメッセージが返される。

エラーの場合は図 16 の様に表示されるが、内部では PMS からクライアントへメッセージ構造体内にエラー番号を代入して返されている。

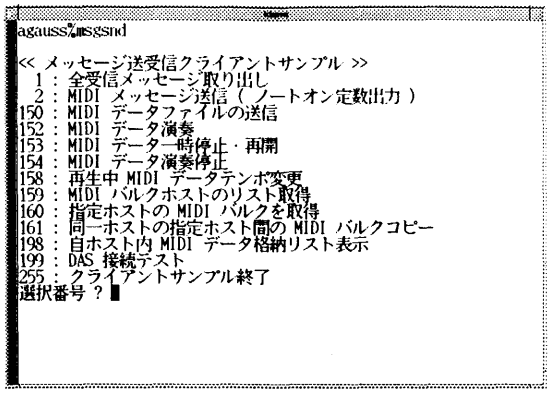


図 14: サンプルクライアントの実行

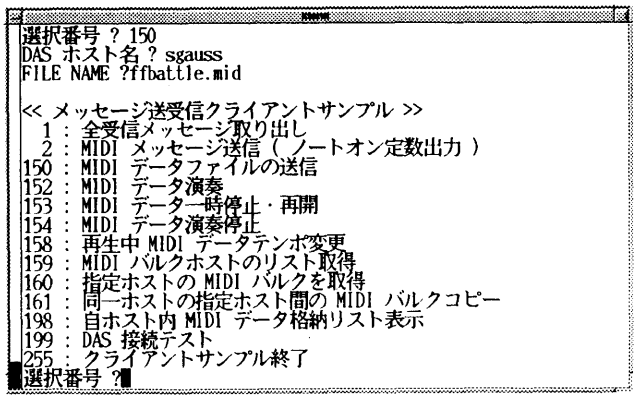


図 15: MIDI ファイルの送信

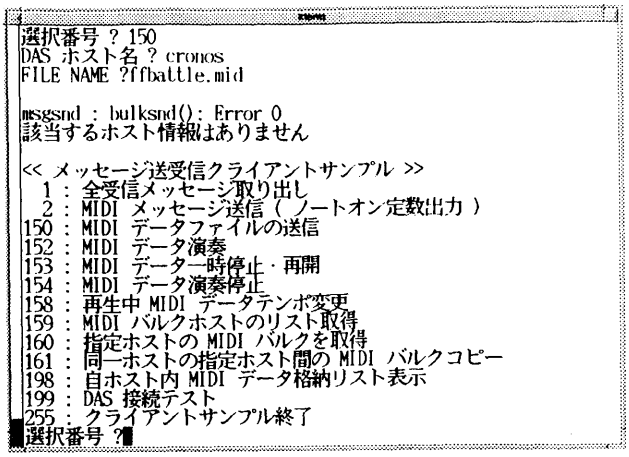


図 16: DAS が起動していないホスト指定の場合

6.5 MIDI データの再生

クライアントメニュー上で番号 152 を選択する。続いて再生するホスト名を入力することで、指定したホスト内 DAS の MIDI データが MIDI 機器に送信 (再生) される。MIDI ファイル送信時と同様、DAS の起動していないホストの指定では PMS からエラーが返される。図 17 に示す様に再生時、DPS により画面上に MIDI データに関する情報が出力される。

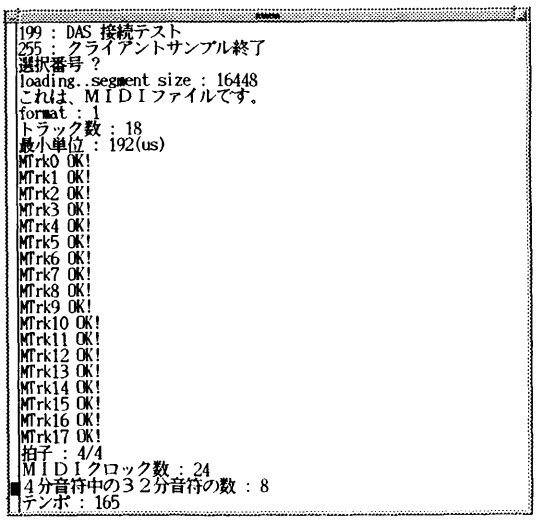


図 17: MIDI データの再生

6.6 MIDI データのコピー

DAS 内部で持つ MIDI データのコピーを行なう機能である。前節で述べた様に DAS は MIDI データをホスト名毎で管理しているが、このコピー機能はあくまで同一 DAS 内でのコピー機能である為、注意が必要である。

図 18 の様にクライアントメニューから番号 161 を選択し、続いてコピー元とコピー先のホスト名を選択することで実行される。後に番号 198 を選択し、コピーが行なわれたかを確認する。

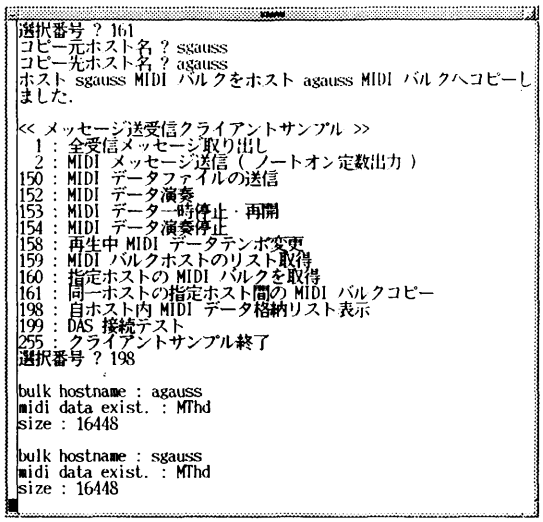


図 18: MIDI データのコピー

6.7 サーバの終了

図 19 の様にサーバの終了は DPS 以外、各サーバのプロセス本体へ SIGTERM シグナル、つまり通常の kill コマンドを起動した順に実行することで行なう。

SIGTERM シグナルを受けたサーバは終了処理に移行し、適切に子プロセスやシステムリソース等の破棄を行ない、メッセージを表示して終了する。

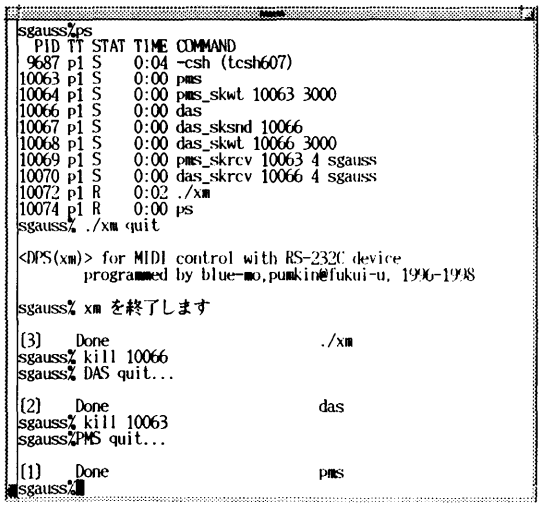


図 19: サーバの終了

7 結論

本研究で開発したMIDI管理サーバによって、複数の遠隔ホストが扱うMIDIデータやメッセージを相互にやり取り出来るようになった。また、昨年度システムのプロセス分散化をより進めた形として、1サーバを複数のプロセスに分離する事により並列処理やサーバの安定した動作を行わせる事が可能となり、MIDIを扱う上での様々な環境に対応し得るシステムが構築出来たと言える。

また、クライアントのメッセージ送受信を簡便にするライブラリを開発した事でMIDI管理サーバの機能を外部から簡単に利用できるため、本システムを用いたMIDIツール、クライアントの開発は非常に容易である。

今後の課題は、ライブラリの充実と更なる安定動作の向上である。データ管理サーバでのメタイベントメッセージの解析や、再生サーバからクライアントへの詳細な再生状況の通知等の機能の追加も必要である。また各サーバのデバッグもより詳細に行うべきである。本研究ではdaemonとして実行する段階にまでは至らなかったが、以上の課題を克服することにより実現可能と思われる。

後の展望は、本システムを用いた、MIDIを教育現場で利用する為の音楽に関する教育支援システムの開発等がある。異なるホスト間のデータ交換が可能となったのでこのようなシステム開発に大きく貢献するであろう。

本研究ではUNIXを開発のプラットホームとしたが、通信機能を持つMIDI制御システムはPC上でも例がない。従って、本システムの他プラットホームへの移植も十分に意義のあるものと言える。

現在ワークステーションのみならずPC上で動作するUNIX互換OSが登場し普及している。本システムをこれら互換OS上で実行させる事が出来れば、高価なワークステーションを用いずとも本システムを導入する事が可能になると思われる。

参考文献

[1] 河野清尊 著「C システムプログラミング入門」オーム社 刊 (1992)

[2] W.Richard.Stevens 著「UNIX ネットワークプログラミング」篠田陽一 訳 (1995)

[3] 出島 豊樹 著: 平成7年度卒業論文「ワークステーションを用いたMIDI制御システムの作成及び作曲・演奏環境の充実」(1996)

[4] 朝井 祥隆: 平成8年度卒業論文「MIDI制御ドライバ及び周辺ライブラリの開発」(1997)

[5] 青山 大介: 平成9年度卒業論文「UNIXによるMIDI Daemonの開発」(1998)